Pictures of a page

The same page, drawn instead of printed: a PNG, JPEG, WebP or TIFF. The page is loaded and its scripts run as for a PDF; then it is drawn as a screen shows it, or printed as the PDF is and drawn a page an image. Since 2.0.0-beta.10, in every package.

What is drawn

Capture	The image
The whole page (the default)	The document as a screen shows it, in one image: laid out in a window of 800 by 600 CSS px in screen media, the image as wide and as tall as the document scrolls. Puppeteer's `fullPage: true`.
The window	The window alone, as a screenshot without `fullPage`.
Each printed page	The page printed as the PDF prints it (paper, margins, print media, page ranges, headers and footers), an image a page.

- **Formats.** PNG (the default, lossless), JPEG and WebP at a quality from 0 to 100 (92 when none is given), TIFF (lossless, one file holding every image).
- **The window.** Its width and height, 1 to 16,384 CSS px: what the page is laid out at (its media queries, viewport units, its scripts' `innerWidth`).
- **Sharpness.** The device scale factor is image pixels per CSS px: 1 draws at 96 dpi, 2 at 192 for a high density screen, 3.125 at 300 for print (0.1 to 10). It sharpens the image only: the page is laid out as at 1, and its scripts and media queries see a device pixel ratio of 1.
- **Size.** An image past 16,384 pixels a side or 100 million in all is cut (a whole page) or not drawn (a printed page), and a note says so.
- **The page's options.** The base address, scripts, the network, consent boxes and advertisements, styles, the locale and time zone, fonts and limits are the PDF's. A screen capture always draws backgrounds and takes neither paper, margins, scale, page ranges nor headers and footers.
- **The licence.** Without a key, pictures carry the trial watermark, as a PDF's pages do.

.NET

`HtmlToImage` sits beside `HtmlToPdf`, with the same `Render`, `RenderFile` and `RenderUrl` and their `Async` twins. `HtmlImageOptions` holds `Capture`, `Type`, `Quality`, `ViewportWidth`, `ViewportHeight`, `DeviceScaleFactor` and `Page`, the PDF's own options. The result's `Images` are the files' bytes, one for a screen capture, one a printed page.

```
using WeblitzPdf;

// The whole page, sharp on a high density screen
var shot = HtmlToImage.RenderUrl(new Uri("https://example.com"), new HtmlImageOptions
{
    ViewportWidth = 1280,
    DeviceScaleFactor = 2,
});
File.WriteAllBytes("example.png", shot.Images[0]);

// Each printed page of an A4 document as a JPEG
var pages = HtmlToImage.RenderFile("templates/letters.html", new HtmlImageOptions
{
    Capture = HtmlImageCapture.Pages,
    Type = HtmlImageType.Jpeg,
    Quality = 85,
    Page = new HtmlPdfOptions { Format = "A4" },
});
for (var i = 0; i < pages.Images.Count; i++)
    File.WriteAllBytes($"page-{i + 1}.jpg", pages.Images[i]);
```

The result also has `PageCount`, `Notes` (what was left out or could not be drawn), `Completed` (false when a limit or the script timeout stopped the render), `Elapsed` and, for an address, `HttpStatus`.

Node.js

`output` names the format, with Puppeteer's screenshot options: `capture: 'viewport'` (or `fullPage: false`) for the window, `capture: 'pages'` for the printed pages, answered as a ZIP of `page-1.png`, `page-2.png`… (a TIFF holds them all in one file).

```
import { renderUrl, renderHtml } from 'weblitzpdf';

const shot = await renderUrl('https://example.com', { output: 'png', viewport: { width:
1280, height: 720, deviceScaleFactor: 2 }, path: 'example.png' });
const pages = await renderHtml(html, { output: 'jpeg', quality: 85, capture: 'pages',
format: 'A4' });   // pages.file is a ZIP
```

Python

```
import weblitz

weblitz.render_url("https://example.com", output="png",
                   viewport={"width": 1280, "device_scale_factor": 2},
path="example.png")
pages = weblitz.render_html(html, output="jpeg", quality=85, capture="pages",
format="A4")   # pages.file is a ZIP
```

The names are Playwright's: `capture="viewport"` or `full_page=False` for the window.

Java

```
import com.weblitzpdf.*;

Weblitz.renderUrl("https://example.com", new HtmlOptions().setOutput("png")
        .setViewport(new Viewport(1280,
720).setDeviceScaleFactor(2))).save(Path.of("example.png"));
Result pages = Weblitz.renderHtml(html, new
HtmlOptions().setOutput("jpeg").setQuality(85)
        .setCapture("pages").setFormat("A4"));   // pages.file() is a ZIP
```

The names are Playwright's: `setCapture("viewport")` or `setFullPage(false)` for the window.

HTTP

```
curl -H 'Content-Type: application/json' \
     -d '{"url": "https://example.com", "output": "png", "options": {"viewport":
{"width": 1280, "deviceScaleFactor": 2}}}' \
     http://localhost:8080/v1/html -o example.png
```

`/v1/html` of the server takes `output` `png`, `jpeg`, `webp` or `tiff`, and `capture`, `quality` and `viewport` in its `options`; printed pages answer a ZIP.